Robert Carlsen
CS 536-1 HW3
4/19/2012

CS 536 Homework 3

We keep track of students' grades in a file with the following format:

- The file contains one or more student records.
- Each student record has one student's name, then their ID number, then zero or more grades, separated by commas.
- Each grade is an integer value, followed by zero or more stars (to represent the number of late days).

Here's a CFG for my grade-file format:

```
file      → record tail
tail      → epsilon | file
record    → NAME IDNUM optgrades
optgrades → epsilon | grades
grades    → onegrade | onegrade COMMA grades
onegrade  → INTLIT optlate
optlate   → epsilon | stars
stars     → STAR | stars STAR
```

nonterminal	First Set	Follow Set
file	NAME	EOF
tail	epsilon, NAME	EOF
record	NAME	EOF, NAME
optgrades	epsilon, INTLIT	EOF, NAME
grades	INTLIT	EOF, NAME
onegrade	INTLIT	EOF, NAME, COMMA
optlate	epsilon, STAR	EOF, NAME, COMMA
stars	STAR	EOF, NAME, COMMA, STAR
record tail	NAME	
NAME IDNUM optgrades	NAME	

onegrade COMMA grades	INTLIT	
INTLIT optlate	INTLIT	
STAR	STAR	
stars STAR	STAR	

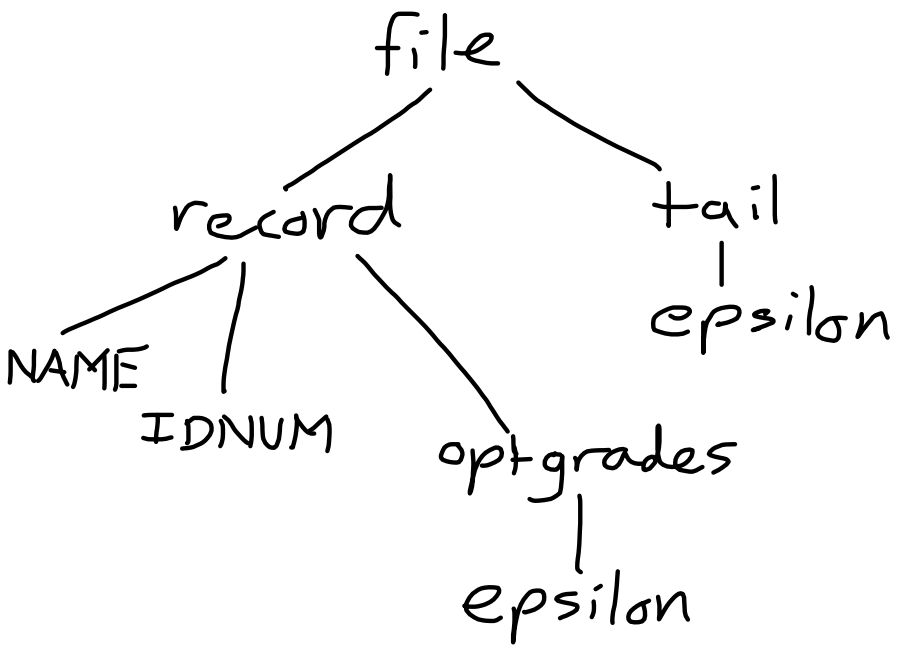
Question 1

Although the grade-file CFG given above is not LL(1), some correct inputs can be parsed by a predictive parser. This is because those inputs never cause the parser to look at a table entry that contains two or more CFG rules.

a. Give the shortest such input (as a sequence of tokens, ending with EOF):

```
NAME IDNUM EOF
```

b. Draw the parse tree that the parser would build for the input you gave for Part (a). (Do not include the EOF token in the parse tree.):



c. Give one more (longer) input that is in the language of the grade-file CFG and that can be parsed by a predictive parser:

```
NAME IDNUM NAME IDNUM EOF
```

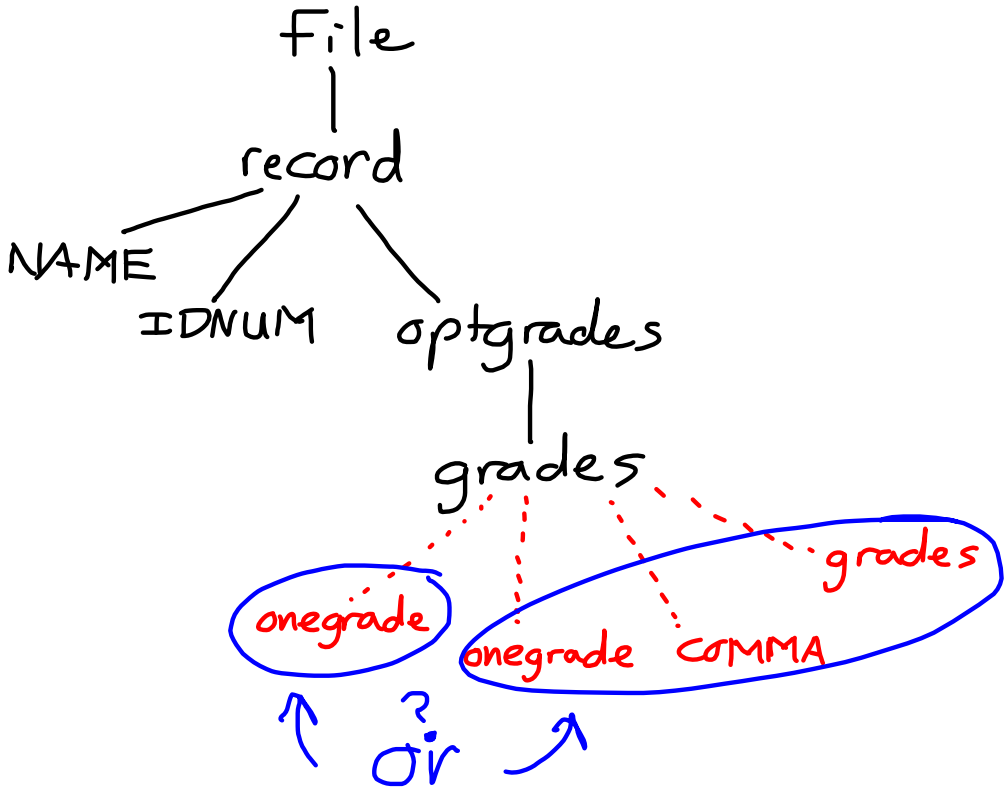
Question 2

What is the shortest sequence of tokens that would "stump" a predictive parser trying to parse the language of this CFG (i.e., a sequence of tokens that is a prefix of a valid input, but for which the parser would not know how to continue to build the parse tree top-down because it looks at a table entry that contains two or more CFG rules)? To answer this question, give all of the following:

1. The sequence of tokens (a prefix of a valid input):

```
NAME IDNUM INTLIT
```

2. The (partial) parse tree that the predictive parser would have built before being stumped.



3. The CFG rules that the predictive parser can't choose between to continue to grow the parse tree:

```
grades → onegrade
grades → onegrade COMMA grades
```

Question 3

One problem with the grade-file CFG is that it has not been left factored. Find the CFG rules with a common prefix, and transform them by doing left factoring:

```
from

grades → onegrade
grades → onegrade COMMA grades

to

grades → onegrade more
more → epsilon
more → COMMA grades
```

Question 4

Another problem with the grade-file CFG is that is has immediate left recursion. Find the CFG rules that cause this, and transform them to remove the left recursion:

```
from

stars → STAR
stars → stars STAR

to

stars → STAR morestars
morestars → epsilon
morestars → stars
```